An Objective Approach to Trajectory Mapping through Simulating Annealing

Stephen Gilbert (STEPHEN@PSYCHE.MIT.EDU)
Department of Brain & Cognitive Sciences
MIT E10-120, Cambridge, MA 02139 USA

Abstract

Trajectory Mapping (TM) was introduced in 1995 as a new experimental paradigm and scaling technique. Because only a manual heuristic for processing the data was included, we offer an algorithm based on simulated annealing that combines both a computational approach to processing TM data and a model of the human heuristic used by Richards and Koenderink (1995). We briefly compare the TM approach with MDS and clustering, and then describe the details of the algorithm itself and present relevant several diagnostic measures.

Introduction

Psychologists have often explored how people organize their knowledge about different objects. A variety of experimental paradigms, such as MDS, or Multi-Dimensional Scaling (Shepard, 1962) and hierarchical clustering (Duda & Hart, 1973; Corter, 1996), have been used to try to elicit the features of stimuli that subjects find important.

In 1995 Richards and Koenderink proposed a new scaling technique called Trajectory Mapping (TM). Broadly speaking, TM is an experimental paradigm that asks subjects for judgments about the features of stimuli and then models the subjects' conceptual structure with a connected graph. Richards and Koenderink describe the paradigm generally and give several examples, but do not offer a detailed algorithm for deriving the graphs from the subject data. We hope to offer a complete experimental TM procedure by describing and analyzing such an algorithm. The graphs resulting from this algorithm resemble those of Richards & Koenderink previously made by hand using various heuristics.

Part I: Overview of the Approach

There are 3 stages to TM. The first is the experimental paradigm, i.e. collecting the data. The second is analyzing the data, i.e. turning it into graphs. The third is interpreting the data, i.e. deciding what the trajectories in the graphs imply about the subject's mental representations.

In each TM trial, a pair of stimuli are chosen and designated **A** and **B**. The subject is asked to note some feature that differs across the two and to choose from the remaining stimuli a sample for each of the blank spaces, an extrapolant in each direction and an interpolant. The resulting quintuplet looks something like this: (extrap. – **A** – interp. – **B** – extrap).

In some cases, extrapolating or interpolating may not be appropriate. For these the subject also has the option to choose "not feasible", meaning that the subject feels uncomfortable choosing any feature that links the pair; "not available", meaning that the subject can imagine the perfect stimulus for a slot, but it is not available in the stimulus set; or "dead end", meaning that **A** or **B** represent a limit for feature extrapolation.

Using these quintuplets of subject data, we use an algorithm based on simulated annealing to generate a connected graph or trajectory map. The links in the graph reflect frequent connections found in the quintuplet data. By further analyzing the trajectory map and the data that created it, we can note subgraphs that might represent features in the data. These subgraphs offer a representation that combines features of both metric scaling and clustering; the subgraphs are ordered clusters.

Comparison with MDS and Clustering

To give an idea of the role that TM can play in relation to traditional scaling techniques, we offer an example knowledge domain, analyzed using MDS, hierarchical clustering, and TM. The domain is a set of 10 subway stations in Boston. This domain is relatively abstract; subway stations can be thought of in several contexts. Where are the stops in relation to each other? Where are the stops in relation to the city above them? Are the stops on the same train line? We will illustrate how each of these questions is answered best by a different representation of knowledge, and therefore by a different scaling technique. The subway stations in this example are *Boylston, Central, Charles St., Copley, Downtown Crossing, Harvard, Kendall, Park St., Science Place,* and *South Station*.

Figure 1 is an MDS plot that comes from asking a subject for the "geographic similarity" of each pair of stations, e.g. "How near is this station to that station, on a scale from 1 to 10?" MDS can turn these data into a metric space. The space illustrates quite well the geographic aspect of subway station knowledge, where the stations are in relation to each other. The dimensions of the space could be vaguely described as North-South and East-West, although they appear slightly skewed here, as predicted by Lynch (1960).

Note, however, that this plot tells us nothing about which stops are connected to each other (the "routes"), or where city boundaries lie (the "clusters").

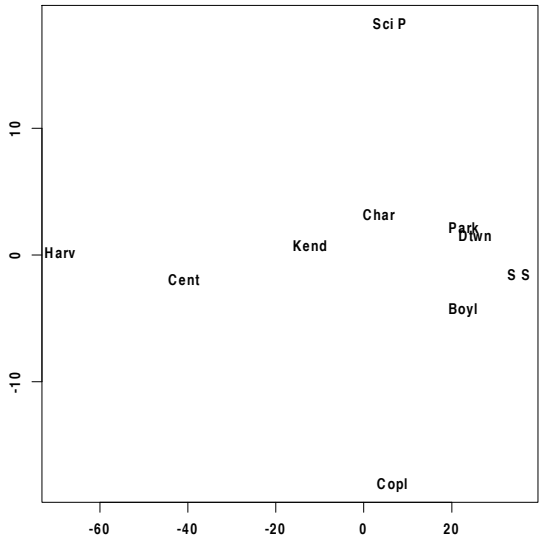

Figure 1: This MDS plot of the similarity data for the subway stations illustrates quite well the geographic nature of subway station knowledge. The dimensions are similar to the North-South-East-West axes of a traditional map of the Boston area.

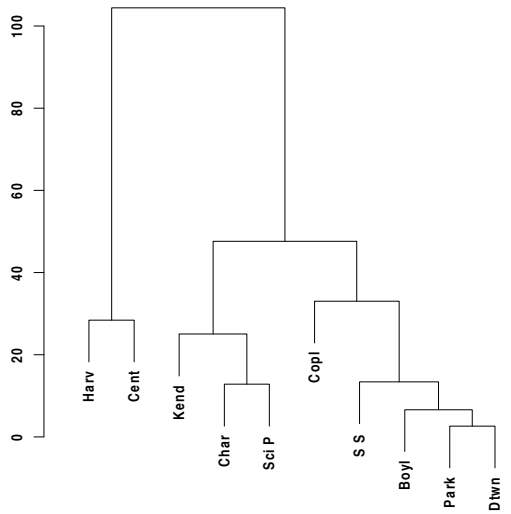

Figure 2: This hierarchical cluster tree of the subway station data reveals clusters based on geographic region, ranging from the core of downtown Boston (right most cluster) to mid-Cambridge (left most cluster).

Figure 2 is the tree of hierarchical clusters representing the same similarity data as Figure 1. Here, the stations can be seen to be clustered by geographic region. The pair of stations in lower right corner represent the core of downtown Boston. The four right-most stations make up a broader downtown area; the five right-most stations are those in main Boston area. The *Kend-Char-Sci P* cluster are those along the Charles river, and the left-most two stops are those deep in Cambridge. These are the groupings that this particular subject makes when he thinks about the subway stops. These clusters answer the question, "Where are the stops in relation to the city?" Note that we still don't know about which stations are on the same lines.

Figure 3 shows the trajectory map of the subway data. By overlaying the orderings of subject data over the graph, we can see which nodes form trajectories. In this case, there are two trajectories that overlap at the *Park* station. This graph reveals the subway lines themselves, something neither of the other approaches could do. The nodes running from *S.S.* to *Harv* are on Boston's Red Line, and *Copl* through *SciP* are on the Green Line. The *Park* node, as it is indeed shown, is an intersection point where riders may change trains. Our clean division of clusters likely stems from the subject mentally moving from station to station along single train lines. If the subject lived at *Copley*, however, and traveled to *Charles St.* for work, then the data would have led us to see *Park* as a node where the paths "turn corners", and we may not have found the division of train lines. Nevertheless, domains with stimuli that can be thought of as varying along separate but overlapping clusters, like subway stations, are very appropriate for TM.

For this domain, each scaling technique contributes a different insight into the data. MDS illustrates where the stops are in geographic relation to each other. The

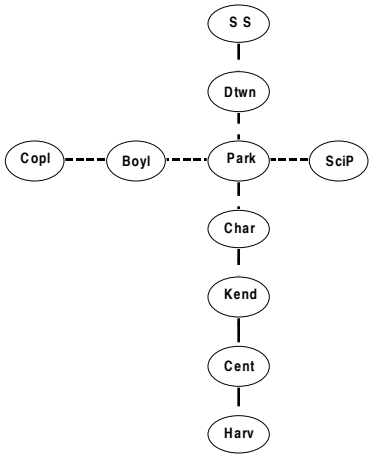

Figure 3: This trajectory map of the subway data reveals the subway lines themselves, something neither of the other approaches could do. The nodes running from *S.S.* to *Harv* are on the Red Line, and *Copl* through *SciP* are on the Green Line.

hierarchical clustering tree groups stops together that have relevance to their area in the city. TM shows which stops are connected to each other, and in which order. This comparison illustrates that each method can plays a part in the analysis and that TM complements the traditional methods well.

It is worth noting that several researchers have proposed using a connected graph representation of similarity data; most prominent are Pathfinder (Cooke, Durso & Schvaneveldt, 1986), NETSCAL (Hutchinson, 1989), and

MAPNET (Klauer, 1989). TM distinguishes itself from these methods by modeling not similarities but clustered orderings of data. By doing so, TM can represent features that are not well expressed by pairwise relations such as similarity.

PART II: The Algorithm

Finding the best graph given the quintuplet data is a combinatorial optimization problem. Richards and Koenderink solve this problem through a manual heuristic that involves systematic trial and error, resulting in what is subsequently taken to be the "simplest" graph, given the data, while allowing for some inconsistent data. We offer a TM algorithm that make this procedure objective, using a simulated annealing paradigm.

The first step in the data analysis is breaking the quintuplets into triples. Of a quintuplet A - B - C - D - E, for example, the triples would be A - B - C, B - C - D, and C - D - E. The number of times each triple occurs in the quintuplet data is assigned as the weight on that triple. The weights are normalized by the maximum possible weight (*n* for *n* stimuli) so that the maximum weight is 1.0. See Figure 4.

The triples are used to build graphs through overlaps. If triple (1, 5, 3) occurs frequently (has a high weight), and triple (5, 3, 4) occurs frequently, then we connect them to form the chain 1 - 5 - 3 - 4. If we could continue to connect triples until we formed a complete graph, then the process would be simple. Often triples conflict, however. There are two main ambiguities that must be resolved in the triple linking process. The first conflict is called a "split-or-fit." If our next heavily weighted triple is (5, 3, 2), for example, we aren't sure of the relationship of nodes 2 and 4. We could have each branch from 3 (splitting the chain), or we could fit them both into the same straight chain, giving less importance to the coherence of one of the triples, e.g. 1 - 5 - 3 - 4 - 2.

The other type of ambiguity is simply called a "conflicting triple," i.e. when two triples suggest different ordering for the same three stimuli. In our example, (5, 4, 3) would be a conflicting triple since it contradicts (5, 3, 4). Conflicting triples can indicate noisy subject data, since a subject who does not behave consistently would generate more conflicting triples, but they can also arise from two stimuli close enough to each other in the feature space that the subject considers the two orderings interchangeable. Usually conflicting triples cannot both be satisfied in a graph, although it sometimes makes sense to include them both by fitting them in a small triangular cycle.

The goal of the algorithm, as described above, is to find the best possible graph as a model of the triples. We begin by constructing an initial unit-link graph with all the links that would be necessary to satisfy all the triples. We then optimally adjust this graph for the given triples and for a certain cost function. This optimization takes place by carrying out simulated annealing using Gibbs sampling (Press, et al, 1988). The state variable that we optimize is a binary link-matrix, stochastically adding and removing links to minimize the cost function. After finding the optimal unit-link graph according to the annealing process, we then

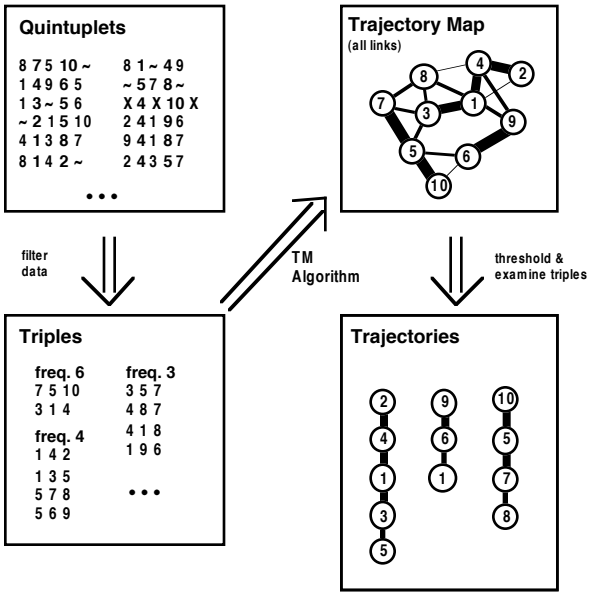
Quintuplets
8 7 5 10 ~ 8 1 ~ 4 9
1 4 9 6 5 ~ 5 7 8 ~
1 3 ~ 5 6 X 4 X 10 X
~ 2 1 5 10 2 4 1 9 6
4 1 3 8 7 9 4 1 8 7
8 1 4 2 ~ 2 4 3 5 7
Trajectory Map
(all links)
filter data
TM Algorithm
threshold & examine triples
Triples
freq. 6
7 5 10
3 1 4
freq. 4
1 4 2
1 3 5
5 7 8
5 6 9
freq. 3
3 5 7
4 8 7
4 1 8
1 9 6
Trajectories

Figure 4: Summary of the Trajectory Mapping procedure.

adjust the parameters of the cost function and begin again. After iterating this process over a large space of cost function parameters, we have a collection of unit-link graphs that are each optimal for their particular parameter settings. (We discuss below how the various parameters affect the optimal graphs.) To calculate the final trajectory map, with a range of weights on the links, we average the optimal unit-link graphs over the space of the cost function parameters. See Figure 5 for an overview of the procedure.

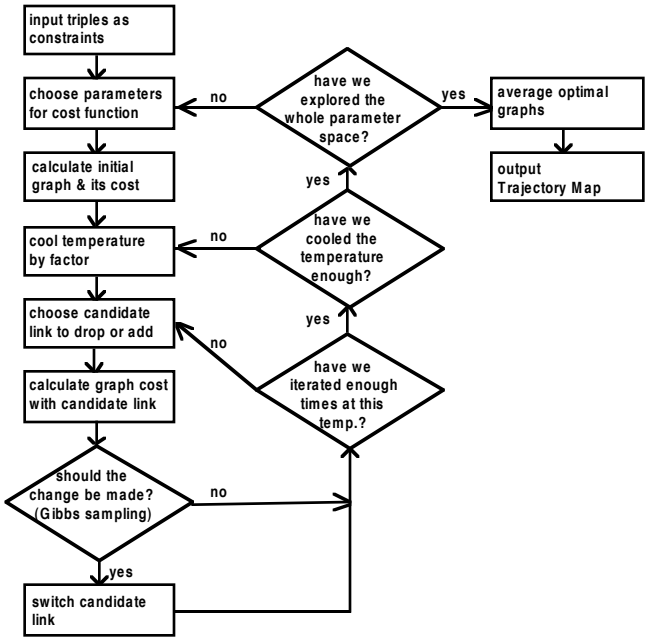
input triples as constraints
choose parameters for cost function
calculate initial graph & its cost
cool temperature by factor
choose candidate link to drop or add
calculate graph cost with candidate link
should the change be made? (Gibbs sampling)
switch candidate link
have we explored the whole parameter space?
have we cooled the temperature enough?
have we iterated enough times at this temp.?
average optimal graphs
output Trajectory Map
no
yes

Figure 5: Overview of the simulated annealing portion of the TM algorithm.

The Cost Function

The cost function is a mathematical model that chooses how to settle the various ambiguities that arise when trying to link triples as described above. Our cost function also represents an attempt to model the manual heuristic of linking triples that Richards and Koenderink (1995) suggest.

The cost function contains five costs, each of which is based on an assumption about the constraining triples. To summarize the cost function, graphs can differ depending on whether the cost parameters emphasize satisfying all the triples (giving a more connected graph) or keeping the graph structure simple (giving a graph with more linear chains and fewer satisfied triples). To break down this issue, we have costs of three types, constraint costs, metric costs, and topological costs.

Constraint Costs A constraint cost is based on the degree to which the constraints in this optimization problem (the triples) are satisfied. For example, if we need to satisfy the triple 1 - 5 - 3 but 3 is closer to 1 than 5 in the current graph, that triple is not satisfied. We have one term in the cost function, called the FailedTriple cost, to penalize for unsatisfied constraints. Because we want to emphasize the priority of satisfying triples with higher weights, the FailedTriple cost is the sum of the logs of the weights of the triples that the graph left unsatisfied. This cost stems from the initial idea that the graph should be a good representation of the triples from the subject data.

Metric Costs There are two terms to penalize for metric costs, the UnequalSpacing cost and the FarSpacing cost. We suggest the graph incurs metric costs if it includes a triple within the graph, but it does not allow the nodes of the triple to be adjacent. If the nodes of a triple are unequally spaced or spaced quite far apart, e.g. 1 - 5 - 3 is satisfied, but the graph distance between 1 and 5 is 1, while the graph distance between 5 and 3 is 4, then the graph has incurred an UnequalSpacing cost. If the nodes are spaced far apart, but are equally spaced, then the graph has incurred a FarSpacing cost. The UnequalSpacing cost for a given triple is the difference between the number of nodes between the three triple nodes, multiplied by the weight of the triple. The FarSpacing cost for a given triple is the number of extra nodes between the triple nodes, again multiplied by the weight of the triple. These metric costs stem from our assumption in TM that when the subject performs the extrapolations and interpolation in the original quintuplet, she would prefer to pick stimuli that result in close, equidistant quintuplets.

Topological Costs There are also two terms to penalize for topological costs within a given graph. The TotalLinks cost is equivalent to a "price" per link; each additional link has a cost. The MaxLinksPerNode cost encourages links to be spread across nodes instead of stemming from just one or two individual nodes by assigning a penalty proportionate to the greatest number of links on any one node (see below). The TotalLinks cost come from the modeling assumption that the simplest graph possible should be used to model the data (Ockham's Razor). The MaxLinksPerNode cost stems from the assumption that it would be rare for one stimulus within a domain to have many more features associated with it than the other stimuli. Thus, the cost function can be expressed as:

$$cost = w_{ft}x_{ft} + w_{us}x_{us} + w_{fs}x_{fs} + w_{tl}x_{tl} + w_{ml}x_{ml}$$

where x_{ft} is the FailedTriple cost, x_{us} is the UnequalSpacing cost, x_{fs} is the FarSpacing cost, x_{tl} is the TotalLinks cost, and x_{ml} is the MaxLinksPerNode cost.

The Parameters of the Cost Function

Each of the five terms includes a parameter w which weights that term in relation to the others. Each vector of parameters $\vec{w}$ defines a graph space in which we can perform simulated annealing. A good way to think about the parameters is as a set of priorities about whether to satisfy triples or cut down on links. See Figure 6. The fewer links there are, the fewer satisfied triples there can be. For example, if the FailedTriple parameter is very low (allowing triples to fail pell-mell) or if the TotalLink parameter is very high (reducing the number of links drastically), then the graph will likely have very few links at all, let alone chains that could make meaningful trajectories. Likewise, if the FailedTriple parameter is very high or the TotalLink parameter is very low, then links will flourish and the graph will be a dense, fully-connected mess.

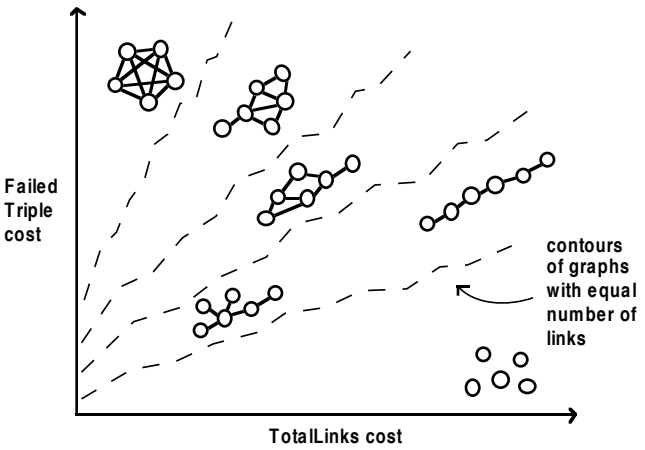

Figure 6: A schematic illustration of the trade-off between satisfying triples and keeping graphs simple.

Thus we can cover a useful area of the parameter space by exploring a three-dimensional subspace of the five-dimensional parameter space by using just the three parameters for FailedTriples, TotalLinks, and FarSpacing. Because graphs can vary so dramatically in this space, the algorithm runs at a wide range of parameter settings and then averages the resulting graphs. "Average" means that if link (1 - 3) occurred in 50% of the graphs in the sampled parameter space, then the output graph has a weight of 5.0 (of 10.0) on link (1 - 3).

The question remains as to how to determine this range of settings. The current sampling of settings has been carefully chosen through trial and error on a variety of different types of data. We run the algorithm over a

sufficiently wide area of space that we are assured that it reaches extremes in all directions. The extrema can be recognized when graphs are fully connected, or when there are no connections at all. The sampling is not completely uniform; at lower values of the parameters we sample more densely, because smaller changes in that range have a significant effect on graphs. We believe that the ideal subspace within the parameter space varies slightly depending on various attributes of the stimulus domain, but our experiments in that area are not yet finished. Even if the ideal subspace were found, results would not change dramatically; all the stronger trajectories would emerge the same. More subtle paths might be able to be identified more accurately, however.

Once we have a graph, the experimenter can see how the graph breaks into trajectories by iteratively thresholding the graph (removing links below a certain weight) and by plotting the data triples over the graph. Thresholding points to the strongest trajectories within the graph, and plotting the actual triples often helps disambiguate intersecting trajectories. Because the graphs that appear through gradually increasing the threshold approximate the graphs that one finds in the weight space as one moves from the areas of very dense graphs to the areas of less dense graphs, the TM output graph can be seen as a single representation of a set of the graphs in the parameter space.

Our algorithm outputs the graph as a text file that is structured to be read by the Dot and Neato graph-drawing software of Bell Labs (Koutsofios & North, 1993; North, 1992).

PART III: Diagnostic measures

As with any data collection and analysis procedure, diagnostic measures are important for answering questions like, "How well does my model fit the data?", "How noisy is the data to begin with?", and "How similar are these two subjects' models?" Below we describe three diagnostic measures created for TM. The first is a simple test for whether a set of triples is random or not. The second measures the explanatory power of the resulting trajectory map. The third provides a method of comparing different trajectories maps of the same data.

Measure of Randomness

It is helpful to have a measure of randomness of the subject data. We measure randomness by comparing the distribution of the triple weights from the subject's data with the distribution of weights that would occur if data were created arbitrarily, i.e. created by hundreds of Monte Carlo simulations of subjects.. If the two distributions differ significantly according to a chi-squared test, then we can conclude that the subject's data is worth examining.

If a TM subject ignored our instructions and answered randomly, the weight distribution of the subject's triples would match the Monte Carlo distribution closely. To measure the significance of the difference, we calculate the z-scores of the differences for each weight level (since the threshold for significance falls exponentially with increasing weight). Even the smallest difference in the distributions at high weight levels such as 7 or 8 results in an enormous z-score. For a Monte Carlo subject, for example, with medium-sized N (N=15), the likelihood of generating a triple with weight greater than 4 is very small ($< 2.4 \times 10^{-5}$), so any subject data that we find with even one triple with a weight of 4 or greater is already likely to be far from random.

Measure of Trajectory Map Fit to the Data

Once we are satisfied with input data, we need a measure of fit, a number which tells us how well the output of the TM algorithm models the data. To calculate measure of fit between a set of triples from a subject and the resulting trajectory map, we make a list of the triples held within the model, and then we compare the two lists. So that we can assign weights to the model-based triples, we first assign costs to the links of the graph that are inversely proportional to the link weights. We can then list the triples contained in the graph with their costs. The cost of a triple consists of the cost of all links traversed while moving from the first node to the third node. Once we order the model-triples, we can compare this list of triples with the original data

We then use two different measures to assess fit, each based on a comparison of these two lists of triples. The first is simply the percentage of unmatched triples, i.e. the percentage of the data triples that were not included in the model. If a model contains 3 out of the 4 data triples, for example, this measure is 25%. This measure does not penalize the model for containing additional triples beyond the data, however, and thus a fully-connected model would satisfy 100% of the triples while offering little insight into the domain. Also, this measure gives us no indication as to whether the weightings on the satisfied triples are appropriate.

Our second measure of fit is based on ranking the triples in the two lists, and then calculating the Kolmogorov-Smirnov statistic, D, (Press et al, 1988; Siegel, 1956) for the two cumulative distributions of the two lists of ranks. D equals the maximum difference between the two distributions. If the model contained exactly the same triples as the data, weighted in the same order, the statistic would be zero. As the model adds additional triples (as it often does just because of the necessary topology of a graph that models other triples), the distribution of model-triple ranks becomes distorted in comparison to the ranks of the data triples. Thus, this measure penalizes for the additional triples in a model that the matched-triple measure does not take into account.

Both of these measures are key for determining the level of "noise" in the data. Because it is likely that lower-weighted triples contain more noise, we threshold the triples data, i.e. remove triples weighted below a certain threshold before using them as input for the algorithm. The measures of fit are used to determine the threshold. We run the algorithm on sets of triples based on all possible thresholds, and then examine the various measures of matched triples and the Kolmogorov-Smirnov statistic. Note that when we measure the number of matched triples, we compare the model-triples with *all* data triples, even if the model was based on only triples of weight 3 and above, etc. Using

both measures, we can see at which threshold the model fits best.

Measure of Similarity of Two Trajectory Maps

Because we have proposed a new model for representing subject data, it is important that we also propose a measure for rating the similarity of two different models. Given two trajectory maps, we look at the lists of the weighted links for both graphs, and reward for common links and punish for distinctive links. To differentiate between graphs with identical topology but differently weighted links, we also penalize for the difference between weights on the common links. The range of this measure is [-1.0, 1.0], where 1.0 implies identical graphs. Using such a measure, we can compare the trajectory maps of two subjects and decide whether they might be using similar features to construct their maps. It is important to note that our feature measure focuses explicitly on individual links, as opposed to overall graph structure.

Summary & Conclusions

We have described an algorithm designed to build trajectory maps from subject data objectively. Based on the simulated annealing, the algorithm uses triples derived from subject data as constraints that can be used to find an optimal connected graph. We have chosen parameters of the cost function so that the algorithm models the manual heuristics followed by Richards & Koenderink (1995). Lastly, we introduced three diagnostic measures for trajectory maps: a measure of subject data noise, a measure of fit for a trajectory map its data, and a measure of similarity between two trajectory maps. We believe that this algorithm offers an useful method of creating trajectory maps that closely mimics the original intentions of Richards & Koenderink.

A more detailed explanation of the algorithm can be found in Gilbert (1997). This work also includes Trajectory Maps of a variety of data sets, including kinship terms, colors, sound textures, musical intervals, Boston tourist attractions, and knowledge representations. The web page at <http://www-bcs.mit.edu/~stephen/tma> allows the reader to download source code for the algorithm.

Acknowledgments

This work was sponsored by NIH Grant T32-GMO7484. Thanks to Whitman Richards and Joshua Tenenbaum for helpful discussion.

References

Cooke N.M., Durso F.T. & Schvaneveldt R.W. (1986) Recall and measures of memory organization. *Journal of Experimental Psychology: Learning, Memory and Cognition,* 12(4), 538-549.

Corter, J.E. (1996) *Tree Models of Similarity and Association*. Sage university paper series: Quantitative applications in the social sciences, no. 07-112. Thousand Oaks, CA: Sage Publications.

Duda, R.O. & Hart, P.E. (1973) *Pattern classification and scene analysis*. New York: John Wiley & Sons.

Gilbert, S.A. (1997) *Mapping mental spaces: How we organize perceptual and cognitive information*. Doctoral dissertation, Department of Brain & Cognitive Sciences, MIT. To obtain a copy, email docs@mit.edu or see http://nimrod.mit.edu/depts/document_services/docs.html.

Hutchinson, J. W. (1989) NETSCAL: A network scaling algorithm for nonsymmetric proximity data. *Psychometrika*, 54(1), 25-51.

Klauer, K. C. (1989) Ordinal network representation: Representing proximities by graphs. *Psychometrika*, 54(4), 737-750.

Koutsofios, E. & North, S. (1993) *Drawing graphs with **dot***: User's Manual. Bell Laboratories. http://www.research.att.com/sw/tools/graphviz/.

Lynch, K. (1960) *Image of the City*. Cambridge, MA: MIT Press.

North, S. (1992) *NEATO User's Guide*. Bell Laboratories. http://www.research.att.com/sw/tools/graphviz/.

Press, W.H., Flannery, B.P., Teukolsky, S.A., & Vetterling, W.T. (1988) *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge: Cambridge University Press.

Richards, W. & Koenderink, J. J. (1995) Trajectory mapping: A new non-metric scaling technique. *Perception*, 24, 1315-1331.

Shepard, R.N. (1962) The analysis of proximities: Multidimensional scaling with an unknown distance function. I & II. *Psychometrika*, 27, 125-140, 219-246.

Siegel, S. (1956) *Nonparametric Statistics: For the Behavioral Sciences*. New York: McGraw-Hill.